Array Data Structure Chapter 6

B.Ramamurthy

Introduction

- We have using variables that are associated with single objects.
- In this discussion we will look at need for representing collection of data and C++ support for the data arrays.

Need for Data structures

- Represent a collection of similar data objects.
  - Example: List of 157 client account numbers
  - Would you represent it as g1, g2, g3,... g157 ?
- Perform a set of operations on each element of collection of data.
  - Example: Input 157 account numbers, compute and output the complete bill for each account.
  - Are you going to do this 157 different times?

  cin>> g1; s1 = BillingFunc (g1); cout <<s1;
- We need efficient representation and manipulation of matrix-like and table-like structures.

Arrays

- An ***array*** is a collection of data which shares a common identifier and data type.
- Individual elements of the array are specified using ***offsets*** referred to as ***subscripts or indices***.
- In C++ the offsets or subscripts always start with 0.

Array data structure

- Array represents a collection of data objects of the **same type**.
- Syntax:
- **Element_type Array_name [Number_of_elements]**

- The identifier **Array_name** describes a collection of array elements, each of which may be used to store data values of type **Element_type**.
- The number of elements in the collection is given by **Number_of_elements** enclosed in brackets [].
- Each element in the collection is referred by their common name the **Array_name** and by a unique index. The index range starts from 0 and extends to Number_of_elements -1.

Defining Arrays

- General form

  data_type array_identifier[*size*];

  The size of the array may be specified by a defined constant or constant literal.

- Example:

  double m[8];

m[0] m[1] m[2] m[3] m[4] m[5] m[6] m[7]

Array Declaration: Examples

a) float X[8];

b) const int buf_size = 256;
   char buffer[buf_size];

c)

int roll_value;

float payroll[roll_value];

invalid : because roll_value is a variable.

Initializing Arrays

- Some arrays need initialization at the time of declaration itself: Example constant arrays representing statistical tables.

int n[5] = {32, 27, 64, 18, 95};

int crt[] = {6, 7, 8};

int B[10] = {0};

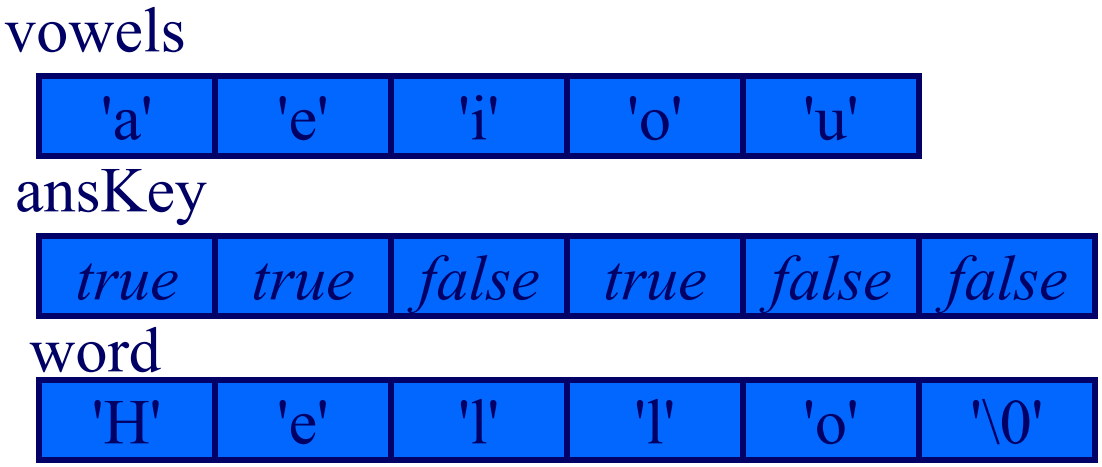
How about int A[4] = {3,4}; ?

A[0] = 3, A[1] = 4, A[2] = 0, A[3] = 0

Initializing Arrays

- Initializing the array when declared

```
char vowels[5] = {'a', 'e', 'i', 'o', 'u'};
bool ansKey[] ={true, true, false, true, false, false};
char word[] = "Hello";
```



Accessing Array Elements

- Subscripts are used to access individual elements of an array.
- General format:

  array_identifier[*offset*]
- Example

```
for (int I=0; I<8; I++)
m[I] = double(I) + 0.5;
```
- Subscripts may be any integer expression.

Array Reference

- How to reference an element of an array ?
- Array_name[subscript] where subscript can be constant, variable or expression.
- Examples:
- cout << X[3];
- cin>> X[5];
- X[6] = 78;
- X[i] = X[j] + X[k];
- X[i] = X[j - 2] + 7;
- if (X[j*2] >= X[i*2/4]) ...

A Simple Array Example

- **Problem 1**: Write a program to input 10 integers, sum them up and output the sum.

```
int main() {
int Sample[10];
int Sum = 0; int i;
cout <<"please input 10 integers\n";
for (i = 0; i < 10; i++)
 {
    cin >> Sample[i];
    Sum = Sum + Sample[i];}
cout<<"the sum = "<<Sum <<endl; }
```

Arrays as parameters

- Array by default is passed by reference in C++.
- The reserved word “const” is placed in front of the prototype and formal declaration in case the array values should remain unchanged during the execution of the function called.

Passing Array as Parameter

- **Problem** : Input two arrays of numbers of 10 elements each, determine their sum in another array, and output the sum array.
- **Design**: Lets discuss the design in terms of function prototypes.

```
void InputAray(int [] X);
// Inputs values into array X
void ArraySum(const int [] A, const int [] B, int [] C);
// Computes C = A+B
void OutputSum(int [] Y);
// Prints out the values in array Y
```

Multi-dimensional Arrays

- To represent tables of data and for modeling activities such as fluid flow and heat flow we need more then one dimensional arrays.
- Multiple-subscripted arrays are used represent multidimensional arrays.
- Example: int b[2][2];
- int b[2][2] = {{1,2}, {3,4}};
- int b[2][2] = {{1}, {3,4}};

...Arrays

- The elements are stored in row-major order.
- When specifying the multi-subscripted arrays as parameters, number of elements in every dimension other than the first has to be specified.

Example (contd.)

```
void PrintArray( int[10][20] a)
{
 for (int i = 0; i <= 10; i++)
 {
   for (int j = 0; j <= 20; j++)
    cout << a[i][j] << ' ';
  cout << endl;
  }
}
```

Functions and arrays

- An array identifier references the first element of the array.
- When arrays are passed as arguments to functions, the address of the array is passed, thus by default, arrays are always passed by reference.
  - This means that any changes made to the array is permanent unless you specify it as a const parameter.
- Generally we specify additional parameters with the array to provided information regarding the number of elements in the array.